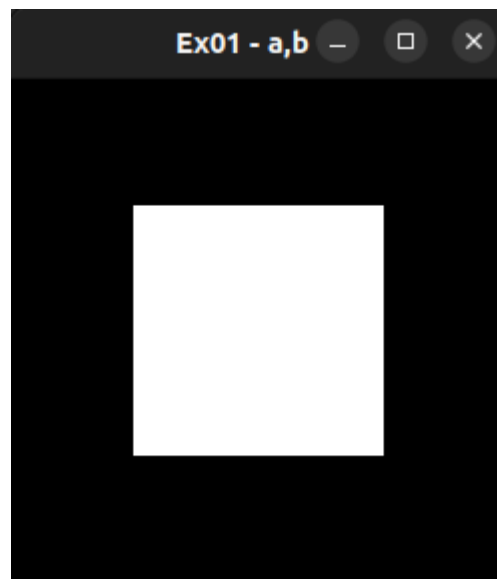


### **Laboratório 02 - Prática com OpenGL**

Através da API gráfica OpenGL e sua estrutura mínima de programa, é possível renderizar um quadrado simples bidimensional utilizando os comandos: `glColor()`, `glVertex()`, `glBegin()`, `glEnd()` e `glFlush()`, passando como argumento as tuplas de vértices e cores, bem como o objeto a ser renderizado ao comando `glBegin()`:

```
glClear (GL_COLOR_BUFFER_BIT);  
glColor3f (1.0, 1.0, 1.0);  
glBegin(GL_POLYGON);  
    glVertex3f (0.25, 0.25, 0.0);  
    glVertex3f (0.75, 0.25, 0.0);  
    glVertex3f (0.75, 0.75, 0.0);  
    glVertex3f (0.25, 0.75, 0.0);  
glEnd();  
glFlush ();
```

Resultado (itens *a* e *b*):

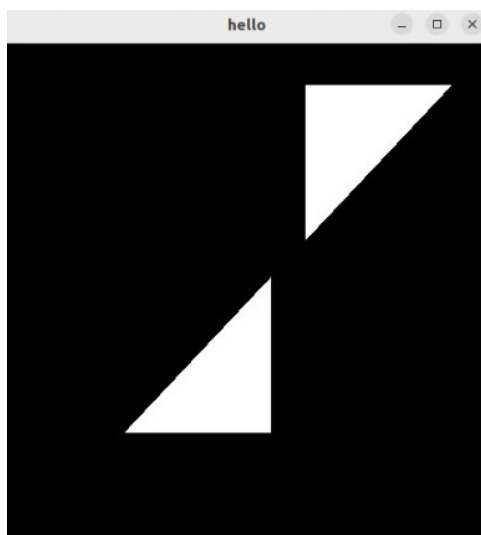
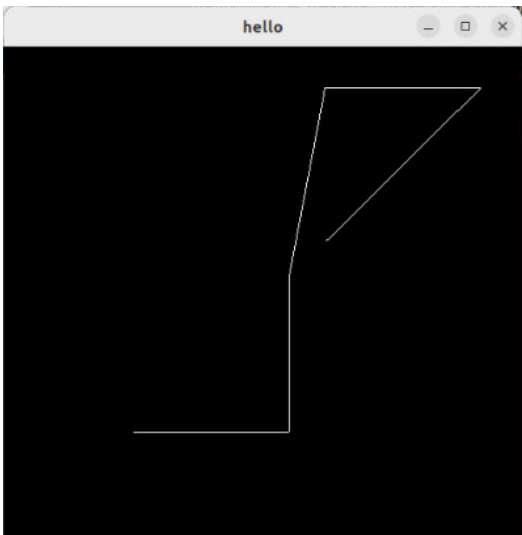
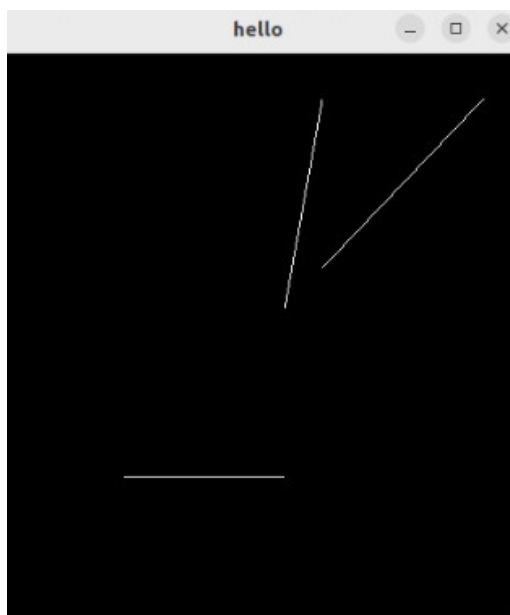


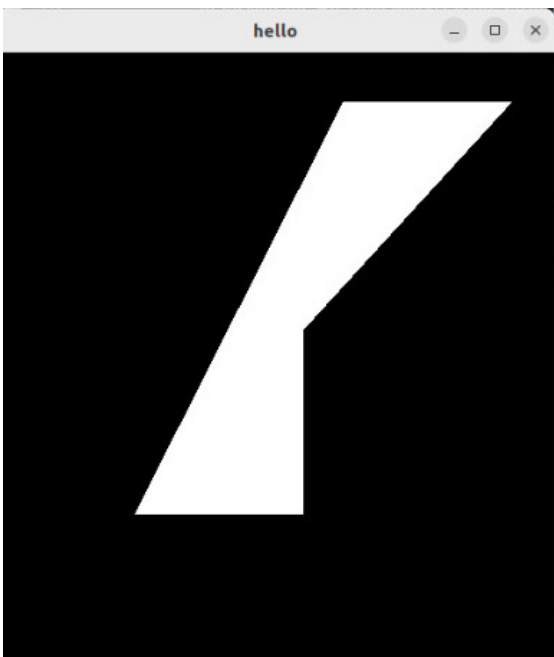
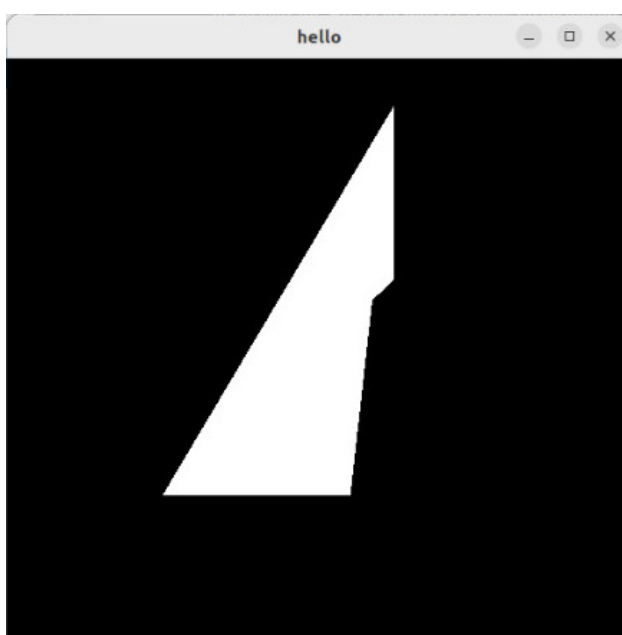
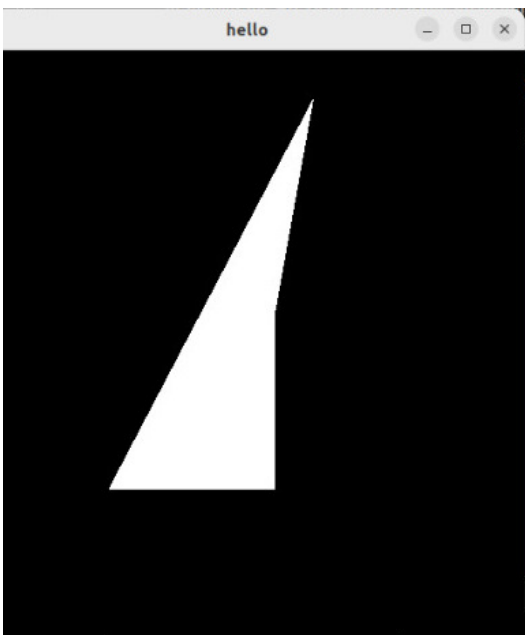
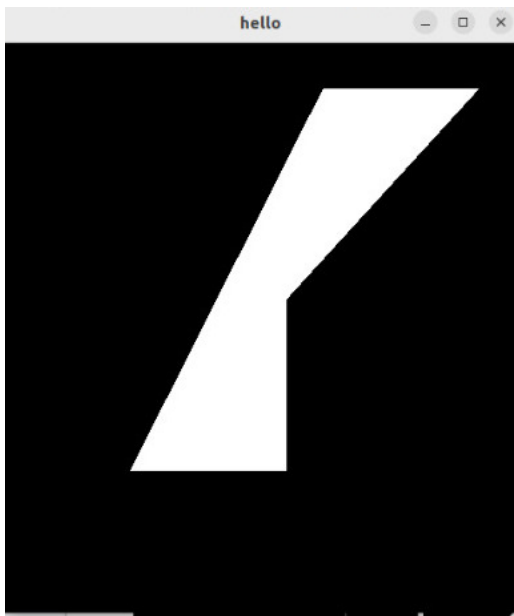
Passando diferentes constantes da API para a função `glBegin()`, é possível gerar diferentes imagens com os mesmos vértices e cores definidos:

### EXERCÍCIO C

Para que ficasse mais claro a diferença entre cada um dos comandos, alteramos o código de maneira a formar dois triângulos, e a partir dessas coordenadas fomos aplicando as primitivas

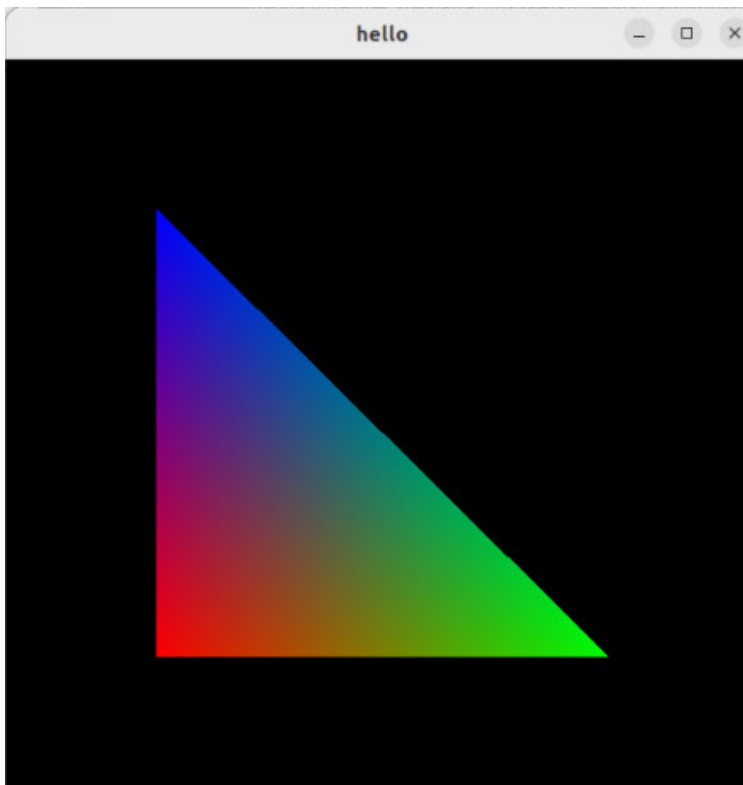
```
glColor3f(1.0, 1.0, 1.0);  
glBegin(GL_LINE_STRIP);  
    glVertex3f(0.25, 0.25, 0.0);  
    glVertex3f(0.55, 0.25, 0.0);  
    glVertex3f(0.55, 0.55, 0.0);  
    glVertex3f(0.62, 0.92, 0.0);  
    glVertex3f(0.92, 0.92, 0.0);  
    glVertex3f(0.62, 0.62, 0.0);  
glEnd();
```



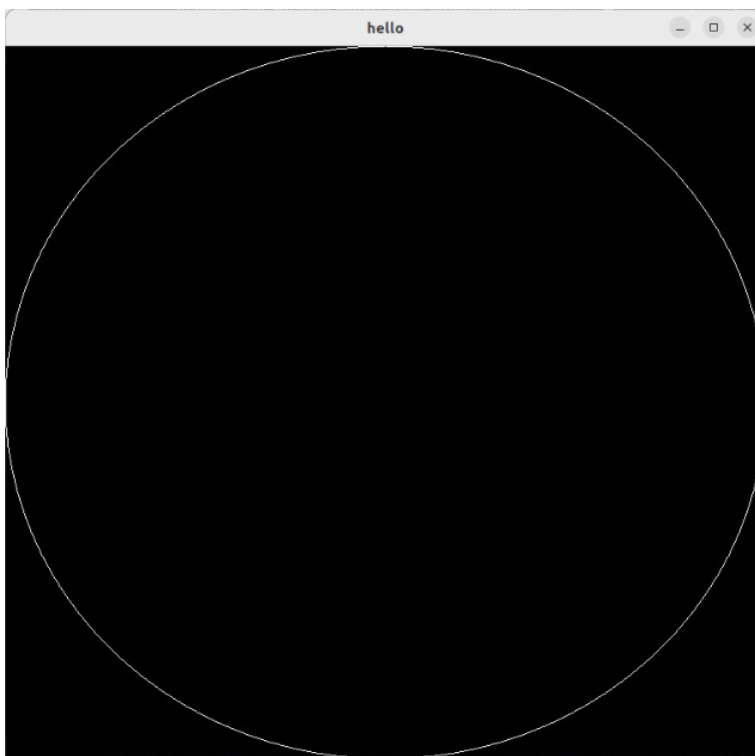


Os resultados acima foram obtidos utilizando as primitivas, GL\_POINTS, GL\_LINES, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_TRIANGLES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_QUADS, GL\_QUAD\_STRIP, GL\_POLYGON, respectivamente.

#### EXERCÍCIO D



#### EXERCÍCIO E



## EXERCÍCIOS PARA ENTREGAR

1-

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include <stdlib.h>
#include <math.h>

// Função callback chamada para fazer o desenho
void Desenha(void)
{
    // Limpa a janela de visualização com a cor de fundo especificada
    glClear(GL_COLOR_BUFFER_BIT);

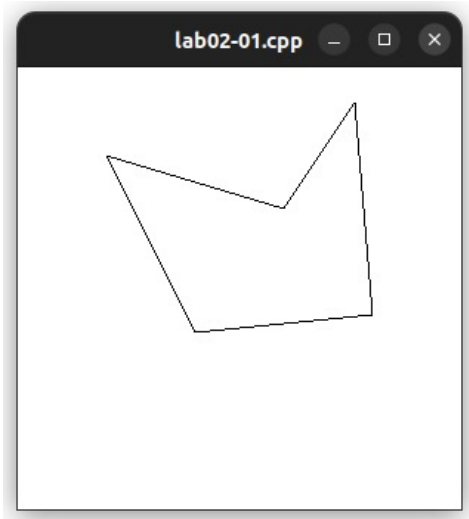
    glBegin(GL_LINE_LOOP);

    glColor3f(0.0, 0.0, 0.0);
    glVertex2f(0.0f, 0.0f);
    glVertex2f(0.25f, -0.5f);
    glVertex2f(0.75f, -0.45f);
    glVertex2f(0.7f, 0.15f);
    glVertex2f(0.5f, -0.15f);
    glEnd();

    // Executa os comandos OpenGL
    glFlush();
}

// Inicializa parâmetros de rendering
void Inicializa(void)
{
    // Define a cor de fundo da janela de visualização como preta
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glOrtho(-0.25, 1, -1, 0.25, -1.0, 1.0);
}

// Programa Principal
int main(int argc, char *argv[])
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutCreateWindow("lab02-01.cpp");
    glutDisplayFunc(Desenha);
    Inicializa();
    glutMainLoop();
    return 0;
}
```



2-

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include <stdlib.h>
#include <math.h>

// Função callback chamada para fazer o desenho
void Desenha(void)
{
    // Limpa a janela de visualização com a cor de fundo especificada
    glClear(GL_COLOR_BUFFER_BIT);

    // glBegin(GL_POLYGON);
    //     glVertex3f(0.25f, 0.25f, 0.0f);
    //     glVertex3f(0.75f, 0.25f, 0.0f);
    //     glVertex3f(0.75f, 0.75f, 0.0f);
    //     glVertex3f(0.25f, 0.75f, 0.0f);
    // glEnd();

    // glBegin(GL_TRIANGLES);
    //     glColor3f(1.f, 0.f, 0.f);
    //     glVertex3f(0.2f, 0.2f, 0.f);
    //     glColor3f(0.f, 1.f, 0.f);
    //     glVertex3f(0.8f, 0.2f, 0.f);
    //     glColor3f(0.f, 0.f, 1.f);
    //     glVertex3f(0.2f, 0.8f, 0.f);
    // glEnd();

    GLdouble PI = 3.1415926535897;
    GLint circle_points = 100;
    GLdouble radius = 1;

    glBegin(GL_LINE_LOOP); // Circulo
```

```

for (int i = 0; i < circle_points; i++)
{
float angle = 2 * PI * i / circle_points;
glColor3f(0.0, 0.0, 0.0);
glVertex2f(radius*cos(angle), radius*sin(angle));
}
glEnd();

```

```

glBegin(GL_POLYGON); //Hexagono
for (int i = 0; i < 360; i += 60 )
{
float angle = i * PI / 180.0;
glColor3f(1.0, 0.0, 0.0);
glVertex2f(radius*cos(angle), radius*sin(angle));
}
glEnd();

```

```

radius = 0.9;

```

```

glBegin(GL_POLYGON); //Quadrado
for (int i = 45; i < 360; i += 90 )
{
float angle = i * PI / 180.0;
glColor3f(0.0, 0.5, 0.0);
glVertex2f(radius*cos(angle), radius*sin(angle));
}
glEnd();

```

```

radius = 0.64;

```

```

glBegin(GL_POLYGON); // Circulo
for (int i = 0; i < circle_points; i++)
{
float angle = 2 * PI * i / circle_points;
glColor3f(0.0, 1.0, 1.0);
glVertex2f(radius*cos(angle), radius*sin(angle));
}
glEnd();

```

```

glBegin(GL_LINES); // Circulo
// for (int i = 0; i < circle_points; i++)
// {
// float angle = 2 * PI * i / circle_points;
glColor3f(0.0, 0.0, 0.0);
glVertex2f(-1.25,0);
glVertex2f(1.25,0);
glVertex2f(0,-1.25);
glVertex2f(0,1.25);
// }
glEnd();

```

```

glBegin(GL_POLYGON); //Triangulo retangulo
for (int i = 135; i < 360; i += 90 )

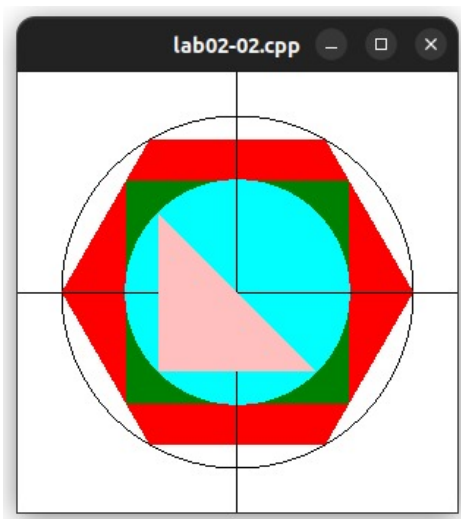
```

```

{
float angle = i * PI / 180.0;
glColor3f(1.0, 0.75, 0.75);
glVertex2f(radius*cos(angle), radius*sin(angle));
}
glEnd();

// Executa os comandos OpenGL
glFlush();
}
// Inicializa parâmetros de rendering
void Inicializa(void)
{
    // Define a cor de fundo da janela de visualização como preta
    // glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glOrtho(-1.25, 1.25, -1.25, 1.25, -1.25, 1.25);
}
// Programa Principal
int main(int argc, char *argv[])
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutCreateWindow("lab02-02.cpp");
    glutDisplayFunc(Desenha);
    Inicializa();
    glutMainLoop();
    return 0;
}

```



### Referências:

- NEIDER, Jackie; DAVIS, Tom; WOO, Mason. **OpenGL programming guide**. Reading, MA: Addison-Wesley, 1993.